Improving experimental design for automatic algorithm selection techniques by using a virtual best solver

H. Degroote, T. Messelis, P. De Causmaecker

KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC*
hans.degroote@kuleuven-kulak.be

Abstract

Evidence of the quality of metaheuristics is usually empirical. Common experimental design consists of testing a technique on benchmark instances. The goal of this article is first, to show that this experimental design is flawed for automatic algorithm selection techniques. Then, to illustrate how experimental design can be improved by calculating a performance upper bound based on the instances and algorithms used. And ultimately, to introduce a tighter upper bound that also takes information about the features into account. The performance upper bounds are calculated using the concept of a virtual best solver, a hypothetical perfect automatic algorithm selector.

Keywords: Metaheuristics, Automatic algorithm selection, Virtual best solver, Experimental design

Metaheuristics are high-level heuristic techniques, applicable to many different problems. Evidence of their quality is usually empirical: a technique is tested on a set of benchmarks and its quality is gauged by comparing its performance to the performance of other techniques on the same benchmarks.

Automatic algorithm selection is a metaheuristic technique with which the strengths of multiple solution strategies can be combined. Implementations of this technique require a set of algorithms, instances, features and a performance measure. The performance of each algorithm-instance pair is calculated and a selection mechanism is initialised using this data and the instances' feature values. This mechanism is then used at runtime to predict which algorithm will perform best on new instances.

*Work supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX. (http://comex.ulb.ac.be)

An automatic algorithm selection technique is useful if it outperforms the best algorithm for a particular problem. Henceforth this algorithm is called the single best solver. However, to prove its merit a technique should outperform other automatic algorithm selection techniques, rather than just the single best solver. Comparing performance with the single best solver measures not only the technique's quality, but also the selected problem's susceptibility to automatic algorithm selection. Consider two example techniques. The first imposes no overhead and always selects the best algorithm to solve an instance with. This technique is called the virtual best solver. It is impossible to create, or even conceive, a better performing technique. Juxtapose this with a second technique that does impose overhead and regularly selects an inferior algorithm. Assume the first technique is tested on a homogeneous set of instances: a set for which the same algorithm is best for every instance. No performance increase is achieved. Now assume the second technique is tested on a heterogeneous set of instances: a set over which the algorithms' dominance is evenly distributed. Despite the overhead and some sub-optimal selections, the second technique is likely to outperform the single best solver. Is the second technique better than the first? Of course not. Incorrect experimental design merely makes it appear so.

A technique's performance is more accurately measured by calculating how much of the maximally achievable performance it can obtain. The measure "%VBS" has been used in this context. It depicts the performance of a technique relative to the virtual best solver's, for a specific experimental environment. An even more informative measure also considers the single best solver's performance and compares a technique's to both.[1]

A virtual best solver does not consider information about features, although their influence on the end result is widely acknowledged to be important. Considering features can tighten the performance upper bound imposed by the virtual best solver in at least one situation. It occurs when two or more distinct instances are mapped to the same feature values, yet are best solved by different algorithms. An automatic algorithm selector cannot distinguish between instances with the same feature values and will map them to the same algorithm. Therefore at least one instance will be mapped to a non-best solver, rendering the performance upper bound formally unreachable. This information is taken into account by the feature-incorporating virtual best solver.[2] It is defined by first grouping all instances with the same feature values and then selecting for each group the single algorithm that best solves it. The feature-incorporating virtual best solver provides a tighter performance upper bound than the virtual best solver. Using it improves the accuracy of the corresponding performance measures.

In this article shortcomings of a naive experimental design for automatic algorithm selection were illustrated. A solution was presented based on the existing concept of a virtual best solver. Ultimately, this virtual best solver concept was extended to also take limitations imposed by the feature set into account.

---

[1]Both measures are based directly on work in "L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. In Theory and Applications of Satisfiability TestingSAT 2012, pages 228241. Springer, 2012."

[2]The feature-incorporating virtual best solver has been formally defined in "H. Degroote, Analysing the quality of an algorithm set in the context of automatic algorithm selection, master thesis 2013-2014"